RSAC 2025 Conference

Many Voices.
One Community.

SESSION ID: IMT-T10

AWS Gone Wild: How Attackers Exploit Your Cloud and How to Outsmart Them

Yonatan Khen
Hunters' Team Axon
linkedin.com/in/yonatan-k/

DISCLAIMER

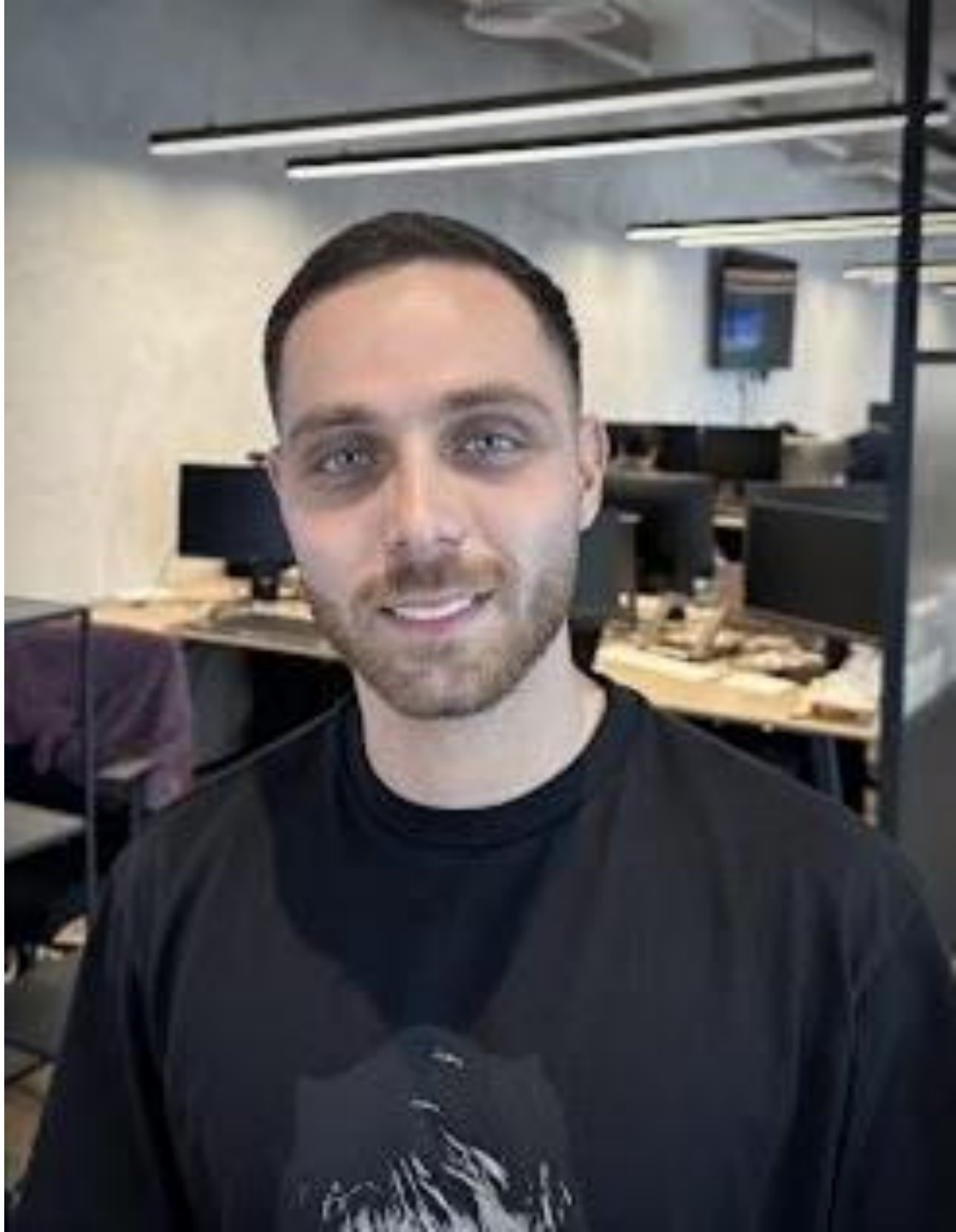
Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference LLC or any other co-sponsors. RSA Conference LLC does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

© 2025 RSA Conference LLC or its affiliates. The RSAC and RSAC CONFERENCE logos and other trademarks are proprietary. All rights reserved.

WHO IS THIS DUDE?!

- 27 year old Security Researcher from Israel
- 8+ years in cybersecurity in the private and public sectors
- Team Axon Lead at Hunters



AGENDA

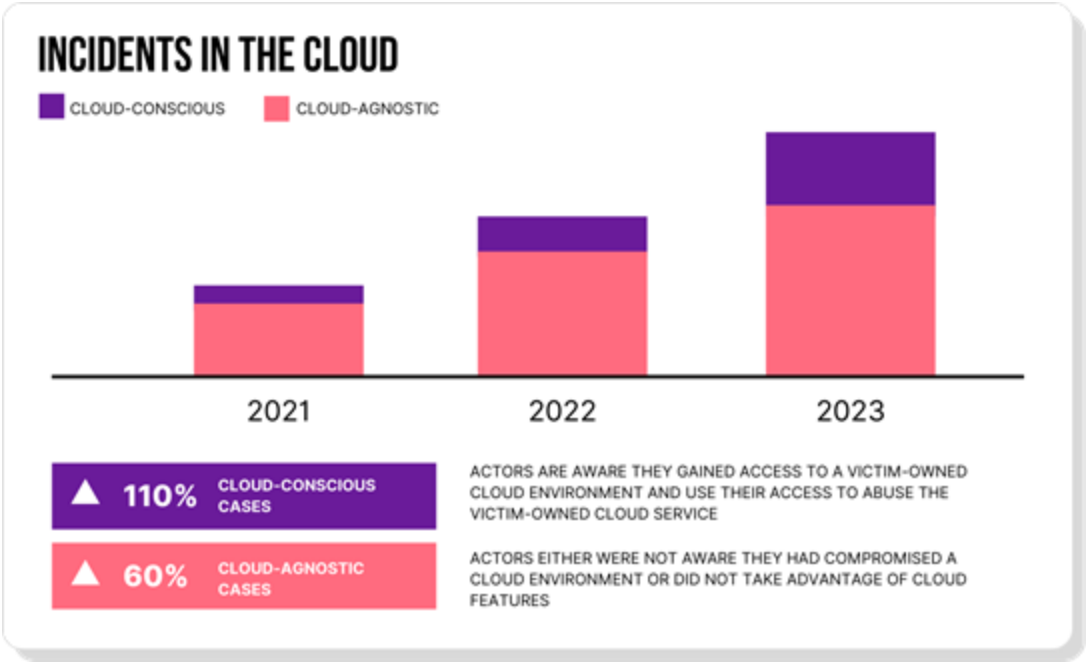
- AWS Cloud Attacks in a Nutshell
- (a real) AWS Incident Overview
- What TTPs adversaries are leveraging in real-world AWS incidents
- Practical Threat Hunting & Detection Ideas

RSAC 2025 Conference

AWS CLOUD ATTACKS IN A NUTSHELL

AWS CLOUD ATTACK

- Cloud attacks have doubled year over year
- Adversaries are becoming increasingly “Cloud-Focused”
- **Not enough** threat publications on cloud attacks are published


INCIDENTS IN THE CLOUD
CLOUD-CONSCIOUS
CLOUD-AGNOSTIC
2021
2022
2023
110%
CLOUD-CONSCIOUS CASES
ACTORS ARE AWARE THEY GAINED ACCESS TO A VICTIM-OWNED CLOUD ENVIRONMENT AND USE THEIR ACCESS TO ABUSE THE VICTIM-OWNED CLOUD SERVICE
60%
CLOUD-AGNOSTIC CASES
ACTORS EITHER WERE NOT AWARE THEY HAD COMPROMISED A CLOUD ENVIRONMENT OR DID NOT TAKE ADVANTAGE OF CLOUD FEATURES

Source: CrowdStrike Global Threat Report

AWS CLOUD ATTACKS IN A NUTSHELL

- Adversaries following the money → Cloud As Target
- The area of “Cloud-Focused” Adversaries
  - Don’t just target “Someone else computer” anymore
  - Understand Control Plane and IAM very well
  - Creative and knows how to evade detections

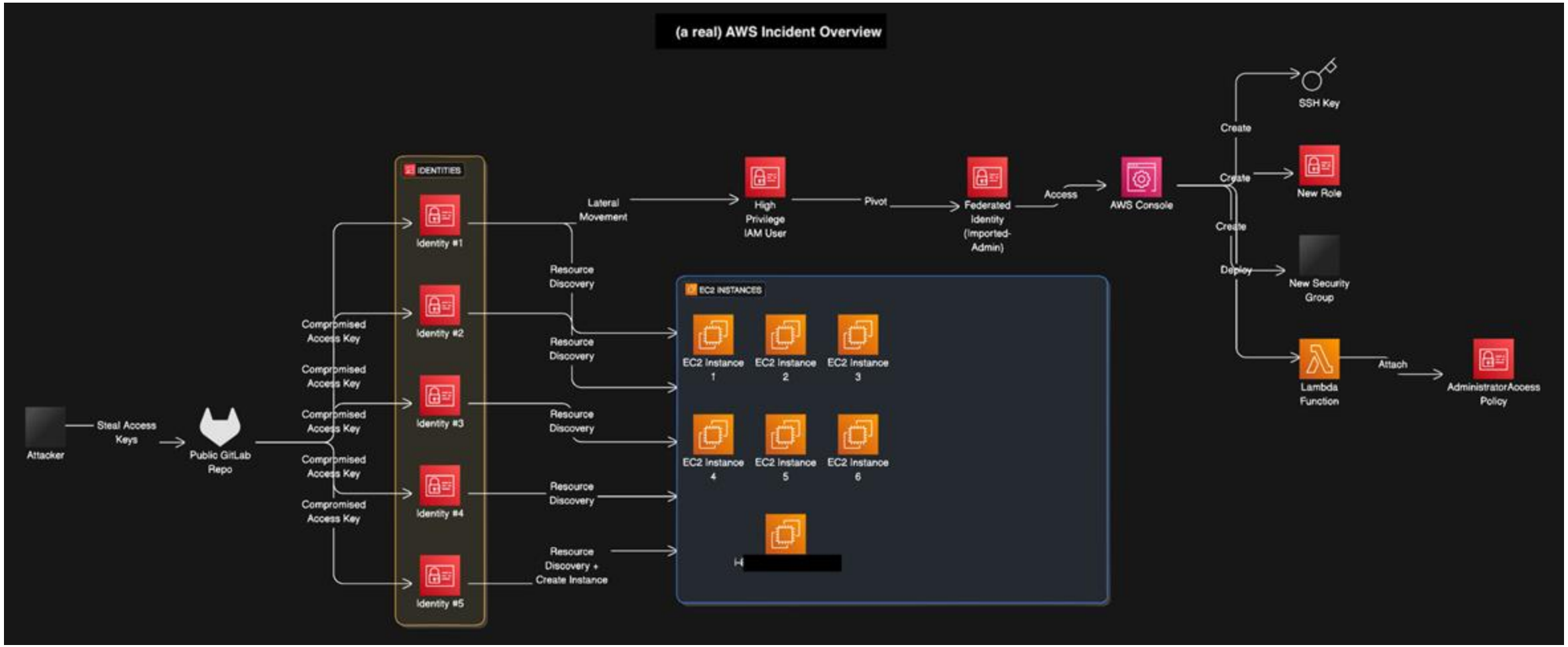
CLOUD ATTACKS IN A NUTSHELL

What TTPs are adversaries leveraging in ***real-world*** *AWS incidents?*

RSAC 2025 Conference

(a real) AWS INCIDENT OVERVIEW

INCIDENT OVERVIEW



INCIDENT OVERVIEW

- One of many similar incidents
- The incident was with objective of accessing cloud resources to steal data, and execute coin miners for instant profit
- The attacker remained in the network for **more than two weeks**
- Insecure coding design allowed the actor to regain access

RSAC 2025 Conference

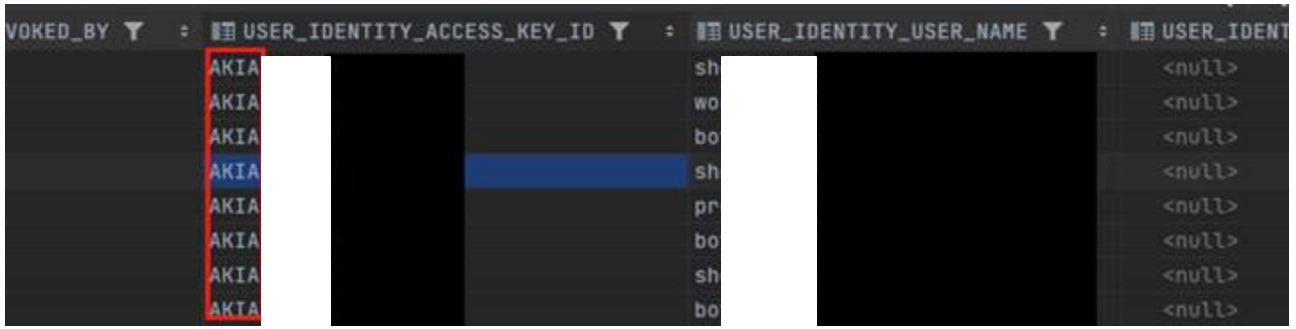
AWS TECHNIQUES IN THE WILD

Discovery Identity Gathering using GetCallerIdentity

AWS TECHNIQUES IN THE WILD

Discovery Identity Gathering using GetCallerIdentity

- *GetCallerIdentity* known to be the classic WHOAMI on AWS attacks by querying AWS *STS* service
- Most incident start with leaked access keys with this technique
- Access key IDs beginning with *AKIA* are long-term credentials for an IAM (compared to *ASIA* which are temporary credentials that are created using AWS *STS* operations)



AWS TECHNIQUES IN THE WILD

Discovery Identity Gathering using GetCallerIdentity

Exposed AWS Access Key → Gather Identity Details on Victim → Post Exploitation API Calls

Adversary scans public repositories and compromise long-term access key

Use GetCallerIdentity API to gather information on target victim

Abuse the credentials to compromise AWS Services

E.G : Trufflehog

AWS TECHNIQUES IN THE WILD

Suspicious GetCallerIdentity Invoked from Unusual Characteristics (Trufflehog)

- Trufflehog is an open source secrets discovery tool that can look for secrets in public repositories
- **HIGHLY** utilized by adversaries in cloud attacks
- Very strong forensic indicator is automatic *GetCallerIdentity* API call the tool performs to validate the discovered Access Key
  - Look for *GetCallerIdentity* API calls with ‘Trufflehog’ UA

```
params := req.URL.Query()
params.Add("Action", "GetCallerIdentity")
params.Add("Version", "2011-06-15")
params.Add("X-Amz-Algorithm", algorithm)
params.Add("X-Amz-Credential", resIDMatch+"/"+credentialScope)
params.Add("X-Amz-Date", amzDate)
params.Add("X-Amz-Expires", "30")
params.Add("X-Amz-SignedHeaders", signedHeaders)

canonicalQuerystring := params.Encode()
payloadHash := aws.GetHash("") // empty payload
canonicalRequest := method + "\n" + canonicalURI + "\n" + canonicalQuerystring + "\n" + canonicalHeaders + "\n" + signedHeaders + "\n" + payloadHash
```

AWS TECHNIQUES IN THE WILD

Suspicious GetCallerIdentity Invoked from Unusual Characteristics

- How To Detect? UEBA methods of first seen set of characteristics (IP, UA, Region) by certain ARN.

Tips:

- Interesting to observe the AWS **region** in the API call
- Exclude AWS IP Ranges (except from EC2 range)
- Exclude Organizational IPs
- AKIA tokens are most likely to be True Positive
- Observe common UA characteristics of access keys “TuppleHug”

AWS TECHNIQUES IN THE WILD

Suspicious GetCallerIdentity Invoked from Unusual Characteristics

RSAC 2025 Conference

AWS TECHNIQUES IN THE WILD

Suspicious Command Executed on EC2 Instance Using SSM

AWS TECHNIQUES IN THE WILD

Suspicious Command Executed on EC2 Instance Using SSM

- AWS SSM is a service that enables you to manage, automate and gain operational insights into AWS resources
- SSM also provides the ability to run shell commands on EC2 instances
- Adversaries leverage this to execute malicious commands on target machines

AWS TECHNIQUES IN THE WILD

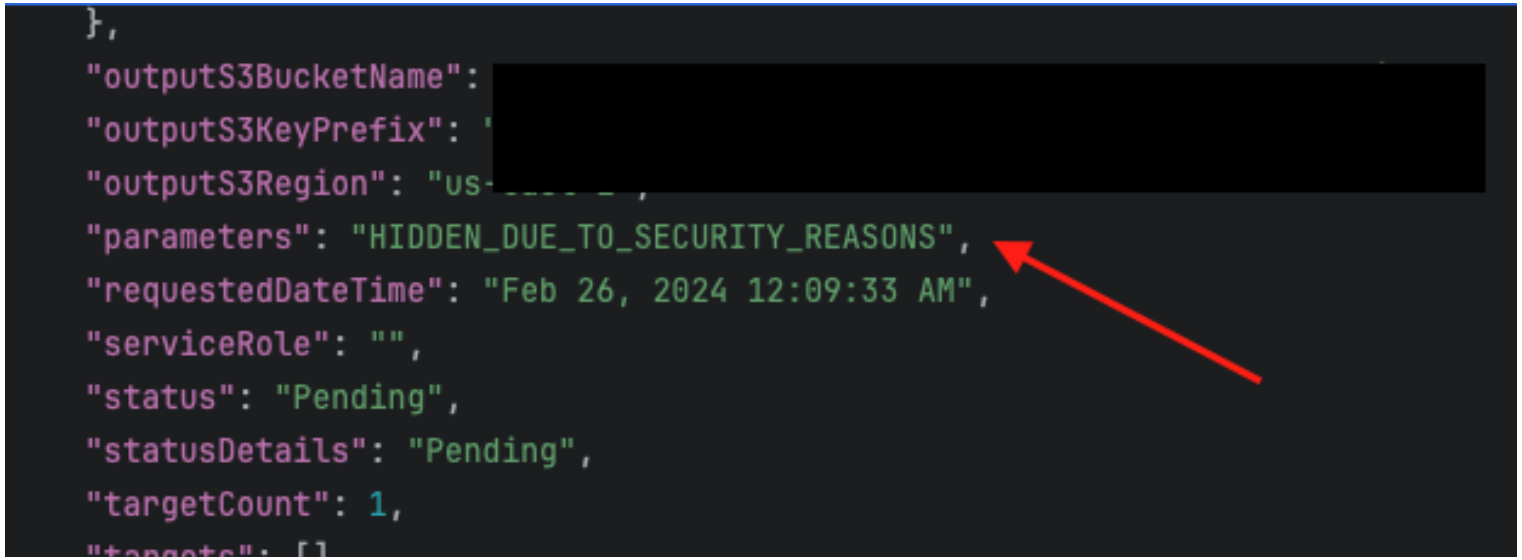
Suspicious Command Executed on EC2 Instance Using SSM

	SendCommand API	StartSession API
Service	SSM	SSM
Target	Multiple instances at once	A single instance at a time
Function	Execute a **remote command** on target instance/s	Create **remote session** via SSM

AWS TECHNIQUES IN THE WILD

Suspicious Command Executed on EC2 Instance Using SSM

- We've explored CloudTrail auditing to understand how to detect unusual activity.
- However, AWS **masks** critical attributes, such as command-line details, limiting visibility



AWS TECHNIQUES IN THE WILD

Suspicious Command Executed on EC2 Instance Using SSM

- *SendCommand* Detection over Process (EDR telemetry) — **HOW?**

Windows

- *SendCommand* executed Under ‘PowerShell’ template
  - Process running under *Powershell.exe* Process
  - Command Line include *awsrunPowerShellScript_script.ps1*

AWS TECHNIQUES IN THE WILD

Suspicious Command Executed on EC2 Instance Using SSM

- *SendCommand* Detection over Process (EDR telemetry) — **HOW?**

Unix

- *SendCommand* executed Under ‘sh’ template
  - Process running under *bash*
  - Command Line include *awsrunShellScript/_script.sh*

AWS TECHNIQUES IN THE WILD

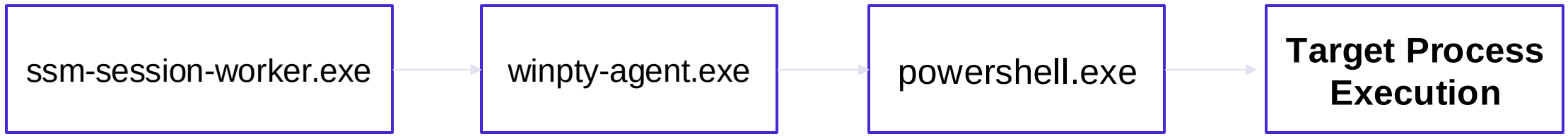
Suspicious Command Executed on EC2 Instance Using SSM

- *StartSession* Detection over Process (EDR telemetry) — **HOW?**

Windows

StartSession executed to initiate a new session

- Grandparent Process Name is *ssm-session-worker.exe winpty-agent.exe*
- Initiating (Parent) Process Name is *powershell.exe*



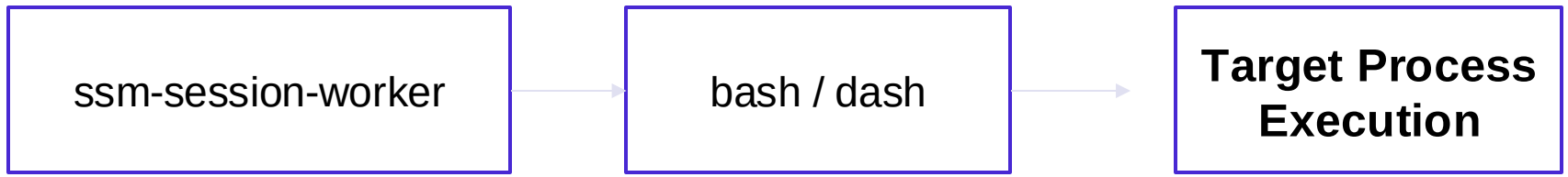
AWS TECHNIQUES IN THE WILD

Suspicious Command Executed on EC2 Instance Using SSM

- *StartSession* Detection over Process (EDR telemetry) — **HOW?**

Unix

- *StartSession* executed to initiate a new session
  - Grandparent Process Name is *ssm-session-worker*
  - Initiating (Parent) Process Name is *bash / dash*



AWS TECHNIQUES IN THE WILD

Suspicious Command Executed on EC2 Instance Using SSM

- **AXON PRO TIPS** (HOW TO **“PRODUCTIONALIZE”**)? AXON

- Try to avoid “Whitelisting” and “Blacklisting” approaches
- Apply UEBA on target execution commands
  - “First time” command characteristics
  - Learning should be applied on “normalized” command line

```
regexp_replace(
  regexp_replace(
    regexp_replace(
      LOWER(target_process_commandline),
      '[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}',     -- Pattern to match UUIDs
      '<uuid>'
    ),
    '[0-9]{1}',                                                          -- Pattern to match every digit
    '0'                                                                  -- Nullify digits
  ),
  '/tmp\\S+',                                                            -- Pattern to match tmp directories
  '<tmp>'
) AS target_process_commandline_normalized
```

AWS TECHNIQUES IN THE WILD

Suspicious Command Executed on EC2 Instance Using SSM

- **AXON PRO TIPS** (HOW TO **"PRODUCTIONALIZE"**)? AXON

- Since *winpty-agent.exe* isn't exclusively spawned by *ssm-session-worker*, we can filter for those executed by *ssm-user*.
  - We highly recommend this tip to minimize the need for multiple JOINs, improving production-ready detection.

- When starting a session on a Linux instance, the *id* and *useradd* commands may run under *ssm-user*
  - Exclude *'%%id %%ssm-user'*
  - Exclude *'%%useradd -m ssm-user'*

AWS TECHNIQUES IN THE WILD

Suspicious Command Executed on EC2 Instance Using SSM

- **AXON PRO TIPs** (HOW TO **“PRODUCTIONALIZE”)?** AXON

- Since *winpty-agent.exe* isn't exclusively spawned by *ssm-session-worker*, we can filter for those executed by *ssm-user*.
  - We highly recommend this tip to minimize the need for multiple JOINs, improving production-ready detection.

- When starting a session on a Linux instance, the *id* and *useradd* commands may run under *ssm-user*
  - Exclude *'%%id %%ssm-user'*
  - Exclude *'%%useradd -m ssm-user'*

RSAC 2025 Conference

AWS TECHNIQUES IN THE WILD

From IAM Creds to Console & Persistence Using Federation Identity

AWS TECHNIQUES IN THE WILD

From IAM Creds to Console & Persistence

- The actor invoked the *GetFederationToken* API to obtain a set of temporary security credentials (user must have the permission *sts:GetFederationToken*)
- *GetFedereationToken* IS NOT *AssumeRole*
- The actor used *GetSessionToken* to generate a temporary token to access the AWS Console without MFA

AWS TECHNIQUES IN THE WILD

From IAM Creds to Console & Persistence

- Temporary credentials issued by *GetFederationToken* are independent of the base IAM user's long-term credentials.
- Disabling or deleting the base user's access **keys does not revoke the federated session for 36 hours**

AWS TECHNIQUES IN THE WILD

From IAM Creds to Console & Persistence

- **How To Detect?** Detect *ConsoleLogin* events with *FederatedUser* Identity Type.

```
-- This hunting query returns all console logins by federated users within the last month
SELECT EVENT_TIME,
       AWS_REGION,
       EVENT_ID,
       RESPONSE_ELEMENTS,
       SOURCE_IP_ADDRESS,
       USER_AGENT,
       USER_IDENTITY_ARN,
       USER_IDENTITY_ACCOUNT_ID,
       USER_IDENTITY_USER_NAME,
       USER_IDENTITY_SESSION_CONTEXT_ATTRIBUTES_MFA_AUTHENTICATED,
       USER_IDENTITY_SESSION_CONTEXT_ATTRIBUTES_CREATION_DATE,
       USER_IDENTITY_SESSION_CONTEXT_SESSION_ISSUER_PRINCIPAL_ID,
       USER_IDENTITY_SESSION_CONTEXT_SESSION_ISSUER_ARN,
       USER_IDENTITY_SESSION_CONTEXT_SESSION_ISSUER_USER_NAME,
       ADDITIONAL_EVENT_DATA
FROM RAW.AWS_CLOUDTRAIL
WHERE EVENT_TIME > CURRENT_TIMESTAMP - INTERVAL '1 month'
AND EVENT_NAME = 'ConsoleLogin'
AND USER_IDENTITY_TYPE = 'FederatedUser';
```

AWS TECHNIQUES IN THE WILD

From IAM Creds to Console & Persistence

- **How To Detect?** Detect *ConsoleLogin* events with *FederatedUser* Identity Type.

Remediation Tip: **Apply Deny all policy on the target IAM**

```
-- This hunting query returns all console logins by federated users within the last month
SELECT EVENT_TIME,
       AWS_REGION,
       EVENT_ID,
       RESPONSE_ELEMENTS,
       SOURCE_IP_ADDRESS,
       USER_AGENT,
       USER_IDENTITY_ARN,
       USER_IDENTITY_ACCOUNT_ID,
       USER_IDENTITY_USER_NAME,
       USER_IDENTITY_SESSION_CONTEXT_ATTRIBUTES_MFA_AUTHENTICATED,
       USER_IDENTITY_SESSION_CONTEXT_ATTRIBUTES_CREATION_DATE,
       USER_IDENTITY_SESSION_CONTEXT_SESSION_ISSUER_PRINCIPAL_ID,
       USER_IDENTITY_SESSION_CONTEXT_SESSION_ISSUER_ARN,
       USER_IDENTITY_SESSION_CONTEXT_SESSION_ISSUER_USER_NAME,
       ADDITIONAL_EVENT_DATA
FROM RAW.AWS_CLOUDTRAIL
WHERE EVENT_TIME > CURRENT_TIMESTAMP - INTERVAL '1 month'
AND EVENT_NAME = 'ConsoleLogin'
AND USER_IDENTITY_TYPE = 'FederatedUser';
```

RSAC 2025 Conference

AWS TECHNIQUES IN THE WILD

Enumeration of multiple EC2 Resources

AWS TECHNIQUES IN THE WILD

Enumeration of multiple EC2 Resources

- Adversaries still tend to target EC2 resources
- A common attack characteristic is enumeration of multiple distinct amount of EC2 resources over a short period of time
- APIs we commonly observe *GetPasswordData SendCommand SendSSHPublicKey StartSession*

Amazon EC2

AWS TECHNIQUES IN THE WILD

Enumeration of multiple EC2 Resources

- **How to detect:**
  - Detect related API calls to multiple EC2 resources over the short time
- **Problem(s):** The data structure of the logs is different from API to another
  - Different key names in the *request_parameters* JSON
  - Some parameters are list (in case the command executed on multiple EC2)

AWS TECHNIQUES IN THE WILD

Enumeration of multiple EC2 Resources

- HOW TO **“PRODUCTIONALIZE”?**

1. Declare all of the relevant keys of the related API calls

```
 CASE
  WHEN request_parameters:targets[0]:key = 'InstanceIds' THEN request_parameters:targets[0]:"values"
  WHEN request_parameters:instanceId IS NOT NULL THEN ARRAY_CONSTRUCT(request_parameters:instanceId)
  WHEN request_parameters:target IS NOT NULL THEN ARRAY_CONSTRUCT(request_parameters:target)
  ELSE request_parameters:instanceIds
END AS array__aws_instance_ids
```

AWS TECHNIQUES IN THE WILD

Enumeration of multiple EC2 Resources

- HOW TO **“PRODUCTIONALIZE”?**

2. Flatten each of the lists

```
SELECT
ct_instance_events.*,
f.VALUE::STRING AS instance_id -- Flattened instance ID
FROM ct_instance_events,
LATERAL FLATTEN(INPUT => array__aws_instance_ids) f
```

AWS TECHNIQUES IN THE WILD

Enumeration of multiple EC2 Resources

- HOW TO **“PRODUCTIONALIZE”?**

3. Apply UEBA / threshold based detection on distinct count of EC2 resources on

RSAC 2025 Conference

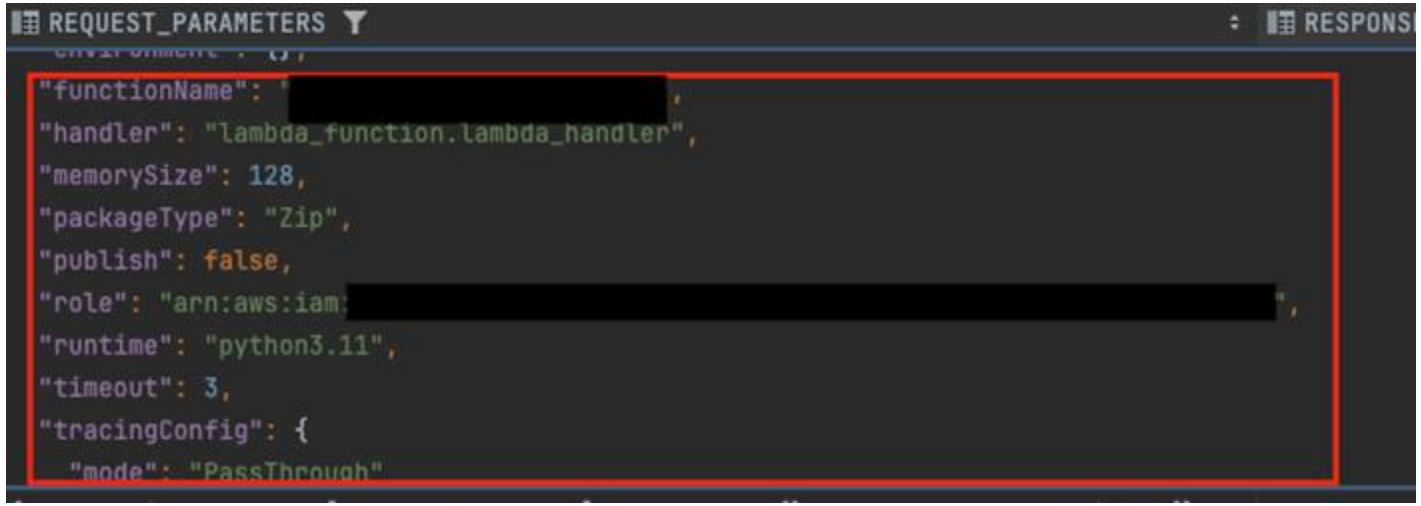
AWS TECHNIQUES IN THE WILD

Persistence Using Lambda Function

AWS TECHNIQUES IN THE WILD

Persistence Using Lambda Function

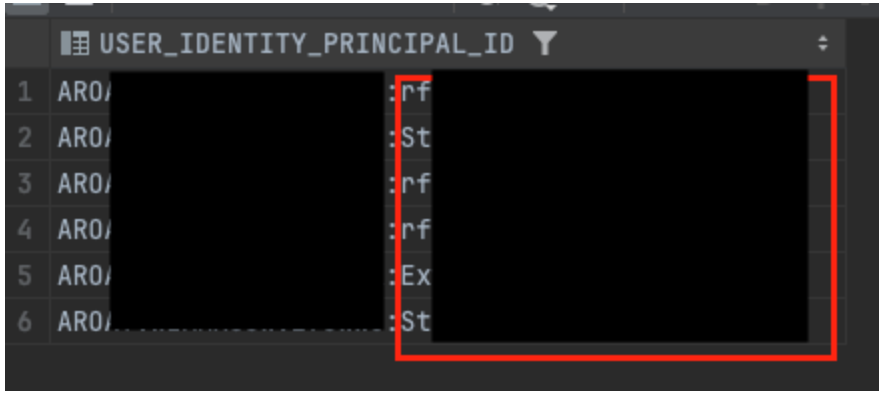
- Lambda is an AWS service for Serverless computing
- Creation of a forged Lambda function can be used for persistence using *CreateFunction20150331* API call
- This is the most used method for persistence in AWS



AWS TECHNIQUES IN THE WILD

Persistence Using Lambda Function

- Problem: How to understand what actions the Lambda function performed?
  - Cloudtrail doesn't provide details to the code
  - Actions made by Lambda performed by AWS, hence, legitimate UA and IP belongs to Amazon subnet.
  - Isn’t shared the same Access Key
- Solution:
  - **Function Name == USER_IDENTITY_PRINCIPAL_ID (Identity ARN Colon)**



RSAC 2025 Conference

AWS TECHNIQUES IN THE WILD

Suspicious Credential Gathering using IMDS

AWS TECHNIQUES IN THE WILD

Suspicious Credential Gathering using IMDS

- The AWS Instance Metadata Service (IMDS) provides instance details, temporary credentials, and network configurations
- Adversaries often abuse IMDS to steal IAM role credential for privilege escalation and lateral movement.
- It can be abused by fetching the IMDS endpoint, with providing the target role
  - *169.254.169.254/latest/meta-data/iam/**security-credentials**/<Target Role>*

AWS TECHNIQUES IN THE WILD

Suspicious Credential Gathering using IMDS

	IMDS V1	IMDS V2
HTTP Method	PUT	PUT & GET
Security Measures	-	- *X-Forwarded-For header* is not allowed - Token enforcement

AWS TECHNIQUES IN THE WILD

Suspicious Credential Gathering using IMDS

HOW TO DETECT? (Process based)

- Detect command lines including *169.254.169.254%security-credentials*
- Filter only use-cases with a string after ***security-credentials*** path
  - In order to fetch credentials, the target role must be specified
- Remove automatic flags *-m, -f, -s*
  - We found those flags together related to automatic activity AWS packages are using

AWS TECHNIQUES IN THE WILD

Suspicious Credential Gathering using IMDS

HOW TO **“PRODUCTIONALIZE”?**

- IMDV2 request includes token in the header *X-aws-ec2-metadata-token*
- To allow better UEBA mechanism and learn on command line keys, normalize the header’s value

```
    TARGET_PROCESS_COMMANDLINE,
    'security-credentials/([^ ]+)',
    1,1,'e')                                    as aws_iam_role_name__role_name,
-- extracting the IAM Role that credentials were accessed for.
  REGEXP_REPLACE(
        TARGET_PROCESS_COMMANDLINE,
        'X-aws-ec2-metadata-token: [A-Za-z0-9=_\-]+',
        'X-aws-ec2-metadata-token: <TOKEN>'
  )                                             as str__normalized_commandline_without_token,
-- IMDSv2 includes a random token flag, this is being normalized so we can run the prevalent values scoring model
```

RSAC 2025 Conference

WRAPPING UP

WRAPPING UP

What we've learned

- Cloud attacks have doubled year over year
- Adversaries are becoming more "Cloud-Focused"
- Despite the increase of Cloud attacks, not enough threat brief publications on Cloud attacks are published.
- What are the most popular attack techniques on AWS among adversaries
- How to practically detect and mitigate them

RSAC 2025 Conference

Many Voices.
One Community.

THANK YOU!
QUESTIONS?

Visit the Hunters at Booth #4514 - Moscone North